Problem of auto-calibration in image mosaicing

Alexey Spizhevoy, Victor Eruchimov
Lobachevsky State University of Nizhni Novogord, Russia
Itseez Ltd., Russia
{alexey.spizhevoy, victor.eruhimov}@itseez.com

Abstract

The paper investigates the auto-calibration problem for mobile device cameras. We extend existing algorithms to get a robust method that computes internal camera parameters given a series of distant objects images. The algorithm is tested on real images generated by several different cameras. We estimate the impact of errors in camera calibration parameters on the image mosaicing problem.

Keywords: *auto-calibration, stitching, camera parameters, errors effect, real datasets, image mosaicing.*

1. INTRODUCTION

The goal of calibration is to determine internal camera parameters within the given projection model. The problem arises in a number of emerging computer vision applications such as augmented reality, 3D reconstruction, and image mosaicing (or stitching). As academy and industry becomes gradually more interested in using mobile devices for computer vision, the importance of phone/tablet cameras calibration is clear.

Nowadays the problem of camera calibration is usually solved by using special calibration patterns (see [3], [4], [5]). While pattern-based methods are quite accurate, it can be difficult to use them due to necessity of taking shots of a special calibration object like a chessboard. Also, manual calibration harms user experience that is considered crucial for mobile applications. As a result software developers and researchers are very interested in auto-calibration methods.

Auto-calibration is the process of estimating internal camera parameters directly from multiple uncalibrated images. This area of computer vision is in active research stage. From one hand there are papers describing successful attempts of using auto-calibration methods in practical tasks (e.g. augmented reality, 3D reconstruction, image mosaics, see [7], [8], [9], [11], [12]). As the topics of these papers aren't camera auto-calibration itself, they don't contain thorough investigations of the used methods with numerical evaluation, tested on challenging dataset. As a consequence, when one faces a computer vision problem that requires camera parameters, it's very difficult to select a robust auto-calibration method and reuse previous results. There is research that is directly devoted to the auto-calibration problem (see [10], [13]). Unfortunately, these papers either don't compare with state-of-the-art pattern-based calibration methods or provide evaluation for synthetic datasets only. Some of these papers describe results for real datasets, but obtained under almost ideal conditions like no noise, no hand shaking, see [13]. So to the best of our knowledge we are not aware of a research paper that describes an auto-calibration method and provides sufficient experimental evidence showing robustness for practical applications.

While classical calibration methods are well studied, they suffer from some drawbacks, which follow from the fact that these methods use some extra information. For instance, there are calibration methods (see [1]) which require location of vanishing points (i.e. points where infinite lines are terminated under projective transformation) as input, but finding of these points automatically is a difficult problem.

This paper shows that under moderate assumptions an autocalibration algorithm for rotational cameras presented in [1] can be used for practical applications with a necessary pre-processing step. We evaluate an implementation of the method for both simulated datasets and real image sequences generated by mobile phone cameras.

2. PROBLEM STATEMENT

We use the following camera model which describes how a 3D scene point $(X, Y, Z)^T$ is projected into an image pixel with coordinates $(u, v)^T$:

$$w \begin{pmatrix} u \\ v \\ 1 \end{pmatrix} = K(R|T) \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix}$$

$$K = \begin{pmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{pmatrix}$$

where K is camera matrix of internal parameters (f_x, f_y are focal lengths in pixels, c_x, c_y are principal point coordinates); R, T are camera rotation 3×3 matrix and translation 3-dimensional vector (external parameters); w is scale factor.

The class of auto-calibration methods that we will consider requires an existence of homography mapping between all input images. The easiest way of generating a sequence of images with homography relashinship using a mobile camera is to take shots of distance objects. Hence, within the scope of this paper we will make an assumption that camera translation T is negligibly small compared to the distance to the objects. We will call a device with $T = 0$ a "rotational camera".

We formulate the auto-calibration problem in the following way: given keypoints in input images taken by a rotational camera, and the keypoint correspondences between images, find the camera matrix K.

3. CAMERA MATRIX ERRORS EFFECT

The estimation of K is never the final goal of a computer vision application. So, in order to understand how precise an auto-calibration method has to be, we need to consider a specific application. This section contains a theoretical and experimental analysis for the image mosaicing problem and provides experimental evaluation on the stitching module of OpenCV library [18]. Throughout this section we make an assumption that f_x equals to f_y for the sake of simplicity and without loss of generality, as images always can be scaled to achieve of unit pixel aspect ratio.

It is possible to stitch images without involving camera matrix. In that case a user wouldn't be able to select another surface for projection except for plane, that can be inappropriate for big panoramas because of big deformations. A plane projection surface generates deformations in the panorama image are visible when the vector of camera orientation differs a lot from the projection plane normal. The most convenient projection surface for the case of rotational cameras is a sphere.

Below we analyze warping errors when the projection surface is a sphere. To compute the error for each image we do the following:

1. For each pixel $q = (x, y, 1)^T$ of the source image we find a ray, passing through the corresponding scene point from camera center, as $r = K^{-1}q$, where K is the camera matrix.

2. We find the intersection point $(X, Y, Z)^T$ of the ray with the unit sphere centered at the origin. This point spherical coordinates u, v after scaling by constant s are point coordinates on the final panorama (s is usually selected being roughly close to the focal length in pixels):

$$u = s \cdot \tan^{-1}(\frac{X}{Z}) \tag{1}$$

$$v = s \cdot (\pi - \cos^{-1}(\frac{Y}{\sqrt{X^2 + Y^2 + Z^2}})) \tag{2}$$

3. To calculate per pixel error we project points using the ground truth camera matrix

$$K^{(gt)} = \begin{pmatrix} f^{(gt)} & 0 & c_x^{(gt)} \\ 0 & f^{(gt)} & c_y^{(gt)} \\ 0 & 0 & 1 \end{pmatrix}$$

and its estimation

$$K^{(est)} = \begin{pmatrix} f^{(gt)} f^{(rel)} & 0 & c_x^{(gt)} c_x^{(rel)} \\ 0 & f^{(gt)} f^{(rel)} & c_y^{(gt)} c_y^{(rel)} \\ 0 & 0 & 1 \end{pmatrix}$$

where $f^{(rel)}, c_x^{(rel)}, c_y^{(rel)}$ are estimated camera parameters relative to the ground truth. The distance between two points obtained using $K^{(gt)}$ and $K^{(est)}$ is the warping error in the pixel p.

According to the presented algorithm we first get two ray directions:

$$r^{(gt)} = \begin{pmatrix} X^{(gt)} \\ Y^{(gt)} \\ Z^{(gt)} \end{pmatrix} = (K^{(gt)})^{-1} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} \tag{3}$$

$$r^{(est)} = \begin{pmatrix} X^{(est)} \\ Y^{(est)} \\ Z^{(est)} \end{pmatrix} = (K^{(est)})^{-1} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} \tag{4}$$

Then we use (1) and (2) to get pixels coordinates $(u^{(gt)}, v^{(gt)})^T$ and $(u^{(est)}, v^{(est)})^T$. The differences between these pixel coordinates are:

$$u_{err} = s(\tan^{-1}(\frac{x - c_x^{(gt)}}{f^{(gt)}}) - \tan^{-1}(\frac{x - c_x^{(est)}}{f^{(est)}}))$$

$$v_{err} = s(\cos^{-1}(\frac{y - c_y^{(gt)}}{\sqrt{(x - c_x^{(gt)})^2 + (y - c_y^{(gt)})^2 + (f^{(gt)})^2}}) + \cos^{-1}(\frac{y - c_y^{(est)}}{\sqrt{(x - c_x^{(est)})^2 + (y - c_y^{(est)})^2 + (f^{(est)})^2}}))$$

The final pixel warp error equals to $\sqrt{u_{err}^2 + v_{err}^2}$. We assess warping errors for the case of 2048×1536 images and using the following camera matrix as a reference:

$$K^{(gt)} = \begin{pmatrix} W + H & 0 & \frac{W}{2} \\ 0 & W + H & \frac{H}{2} \\ 0 & 0 & 1 \end{pmatrix}$$

where W and H are image width and height respectively. The warping error function charts for 5% relative errors in camera internal parameters are shown in figures 1, 2, and 3.

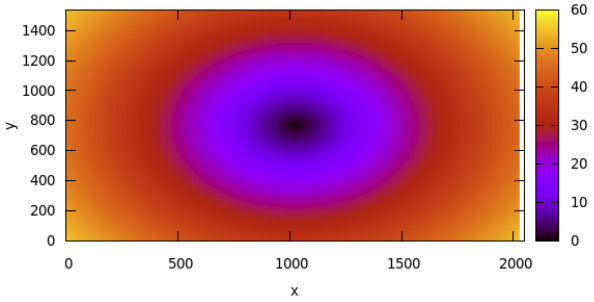

Figure 1: Pixel warp error for $f^{(rel)} = 1.05$.

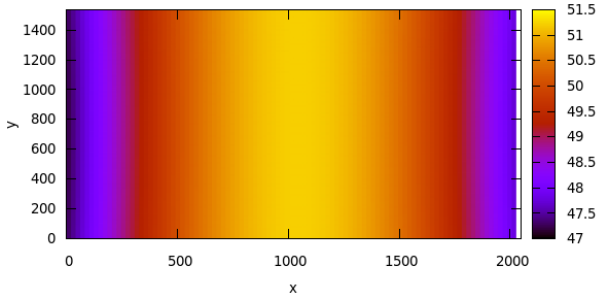

Figure 2: Pixel warp error for $c_x^{(rel)} = 1.05$.

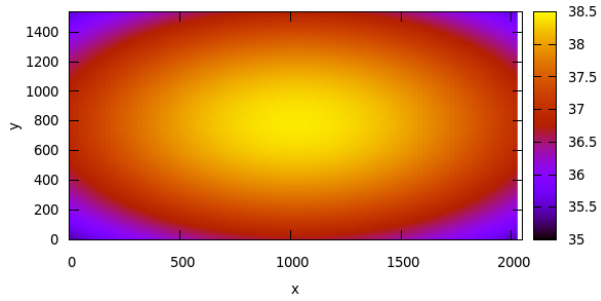

Figure 3: Pixel warp error for $c_y^{(rel)} = 1.05$.

We can see from charts, that when relative error in camera parameters is 5% warp error reaches 60 pixels, that seems to be high enough for leading to visible artifacts.

In order to evaluate the artifacts, we stitched 1536×2048 images using camera matrix $K^{(pt)}$ as ground truth $K^{(gt)}$, where $K^{(pt)}$ was the camera matrix obtained via a pattern based calibration method. Also we did experiments using camera matrix $K^{(est)}$, where each parameter was modified (one at a time) to get 10% error (relative to $K^{(pt)}$). We got panoramas without visible artifacts, see figures 4 and 5. Small artifacts are highlighted with red color, but the quality of the panoramas is much higher that we could expect from theoretical analysis. Such results are obtained because current stitching applications (including the one used for testing) use seam estimation methods to minimize visible artifacts, see [14]. After estimating seams special blending methods are used to hide discrepancies between images, see [15]. So even if the image registration step introduces moderate errors,

a combination of modern seam estimation and blending methods can remove a lot of possible artifacts. But if errors in camera parameters is too high then it's almost impossible to hide stretches and other artifacts, see figure 6 with results for $f^{(rel)} = 0.7$ (i.e. 30% relative error)

Figure 4: Left, right source images and panorama obtained with $K^{(pt)}$.

Figure 5: Panoramas for $f^{(rel)} = 1.1$, $c_x^{(rel)} = 1.1$, and $c_y^{(rel)} = 1.1$ respectively.

Figure 6: Panorama for $f^{(rel)} = 0.7$ with visible artifacts and stretches.

Also it should be mentioned that motions between images are estimated to minimize overall re-projection error (that is minimizing visible mis-registration error) according to the current camera matrix. This step is very important as minimizing re-projection errors leads to minimizing visible artifacts even if the camera matrix was estimated inaccurately. figure 7 shows stitching results of two images with relative motion estimated using $K(pt)$ and with relative motions refined to minimize re-projection errors under assumption that $f^{(est)} = f^{(pt)} f^{(rel)}$ where $f^{(rel)} = 1.1$.

From these results it follows that if one has a high quality stitching algorithm then the effect of errors in camera matrix isn't very high, and methods less accurate than pattern based calibration can be used for camera parameters estimation. This is a good application for auto-calibration that is not as precise as pattern-based calibration but still generates a reasonable estimation of internal camera parameters.

Figure 7: Panorama for $f^{(rel)} = 1.1$ and refined motion vs. panorama for $f^{(rel)} = 1.1$ and motion obtained using $K^{(pt)}$.

4. AUTO-CALIBRATION ALGORITHM

A robust auto-calibration algorithm faces many challenges coming from data generated by a mobile device. Some input images can be noisy, can differ in illumination, and undesired objects such as user fingers can be present in the camera field of view. All these issues can affect the quality of extracted features, and can lead to mis-registration. Hence, let alone the core auto-calibration problem, we have to address these issues. This is why we start with a description of our registration algorithm.

The outputs of the registration algorithm is the images graph, where vertices are images from the input image sequence, and two images are connected with the edge iff we were able to register them with a homography transformation. Here is the description of the registration pipeline:

1. Find keypoints and their descriptors of each image. We use SURF detector and descriptor implemented in OpenCV library, see [16].
2. For each image pair find matches between keypoints. We use FLANN matcher integrated into OpenCV library, see [17].
3. For each image pair estimate 2D homography and compute number of inlier matches, see 4.1.
4. For each image pair determine whether matches between these images are trustworthy, see section 4.1. The decision is made for image pair, not for each match. So if we're confident then we add an edge between two corresponding vertices into images graph.
5. Retain the biggest connected component from the images graph. Also retain only matches for confident image pairs and continue working with this connected component.

4.1 Computing match confidence

We follow the method proposed in [2], where it is applied to extract a subset of images from the original raw set for subsequent stitching.

Suppose we have n_f feature matches. The correctness of an image match is represented by the binary variable $m \in \{0, 1\}$. The event that the i^{th} feature match $f^{(i)} \in \{0, 1\}$ is an inlier/outlier is assumed to be independent Bernoulli event, so the total number of inliers n_i is Binomial. If $m = 1$ then n_i has the $B(n_i; n_f, p_1)$ distribution function, and $B(n_i; n_f, p_0)$ otherwise, where p_1 is

the probability that a feature is an inlier given a correct image match, and p_0 is the probability a feature is an inlier given a false image match.

Here is the final criterion used by the authors to accept an image match

$$\frac{B(n_i; n_f, p_1)P(m=1)}{B(n_i; n_f, p_0)P(m=0)} \geq \frac{p_{min}}{1 \quad p_{min}} \tag{5}$$

Choosing the values for $p_1 = 0.6$, $p_0 = 0.1$, $P(m=1) = 10^{\ 6}$ and $p_{min} = 0.999$ gives the condition

$$n_i > \alpha + \beta n_f \tag{6}$$

for a correct image match, where $\alpha = 8.0$ and $\beta = 0.3$. We decide whether a feature match is an inlier or an outlier by comparing reprojection error with a fixed threshold. We used the same value of 3 pixels for all datasets and that value worked good enough in practice, while for each particular dataset another threshold value can be better.

The value $\frac{n_i}{\alpha+\beta n_f}$ is used as the measure of confidence that it makes sense to use matches between an image pair. If it's greater than 1 then an image match is correct, false otherwise. In some practical cases it could be useful to increase this threshold as was found in experiments.

Figure 8 shows how reprojection error threshold affects on average camera parameters estimation relative error Q for one of real datasets.

$$Q = \frac{1}{4}(|f_x^{(rel)} \quad 1| + |f_y^{(rel)} \quad 1| + |c_x^{(rel)} \quad 1| + |c_y^{(rel)} \quad 1|) \tag{7}$$

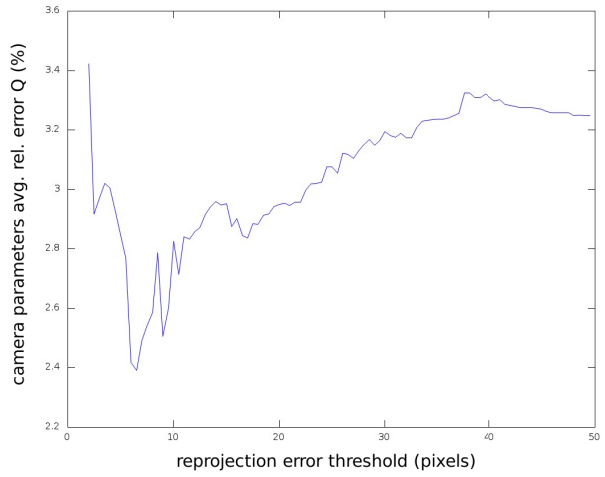

Figure 8: Reprojection error threshold effect on camera parameters estimation errors. When the threshold is too low the algorithm is too sensitive to noise, while in the case of too high threshold even incorrect matches can be classified as inliers.

4.2 Rotational camera auto-calibration

For auto-calibration we use the algorithm for the rotation only cameras case proposed in [1]. Here is the brief description of that algorithm:

1. Normalize the homographies $H_{i,j}$ between views i and j such that $\det H_{i,j} = 1$.

2. Compute $\omega = (KK^T)^{\ 1}$ from the equations

$$\omega = H_{j,i}^T \omega H_{j,i}$$

for all image pairs i, j.

No. of images	Distance (m)	Relative errors (%)			
		$f_x^{(err)}$	$f_y^{(err)}$	$c_x^{(err)}$	$c_y^{(err)}$
6	2	8.5	11.1	4.7	4.6
7	0.5	-3.8	-2.6	-12.4	-6.2
9	2	-3.4	0.1	2.5	5.4
13	2	2.6	7.6	1.5	8.9
14	30	5.6	6.5	-1.9	4.2

Table 1: Relative errors for the auto-calibration of Nokia 6303C camera.

3. Compute K solving $\omega = (KK^T)^{\ 1}$ with the Cholesky decomposition.

4. Refine K by minimizing the re-projection error function

$$err(K, R_1, ..., R_n) = \sum_{i,j,k} \|x_j^{(k)} \quad H_{i,j} x_i^{(k)}\|$$

using parametrization of $H_{i,j} = KR_jR_i^TK^{\ 1}$ over camera rotations R_i, R_j and camera matrix K, where n is the number of images and $x_i^{(k)}, x_j^{(k)}$ are the position of k-th point measured in the i-th and j-th images respectively. We parametrize a rotation with a 3-dimensional vector directed parallel to the rotation axis and with the length equal to the rotation angle.

5. EXPERIMENTS

We performed experiments on real datasets taken with Nokia 6303C mobile phone (1536×2048 resolution) and Logitech QuickCam Pro 900 (1600×1200 resolution).

5.1 Nokia 6303C

Table 1 presents results we got using Nokia 6303C camera. We compare the auto-calibration results with pattern-based calibration: $f_x^{(err)} = f_x^{(rel)} \quad 1 = \frac{f_x^{(est)}}{f_x^{(pt)}} \quad 1$. The auto-calibration algorithm gives relative errors less than 10% on 3 out of 5 datasets. We have showed before that a relative error of less than 10% in camera parameters is enough for getting visually acceptable panoramas.

There are two factors affecting calibration quality. The first factor is the number of images in input dataset, because if the input dataset is too small then it doesn't provide enough information for camera auto-calibration. The second factor is non-zero translation presence, as the auto-calibration method we use was designed under the rotational camera assumption. This assumption is easily violated in practice as a user tends to rotate camera not around its optical center, but around device center (or itself), which is not the same.

5.2 Logitech QuickCam Pro 900

Table 2 presents result we got using Logitech QuickCam Pro 900 camera. For this camera we achieved the relative error less than 9% in comparison with OpenCV pattern based calibration results.

6. TRANSLATION NOISE IMPACT

We also performed experiments on synthetic data to analyze the dependence between camera parameters estimation errors and the translation between camera positions. We created a synthetic scene consisting of 1000 points located randomly on a unit sphere (uniformly in spherical coordinates) centered at the world frame origin and a camera located at the point $(0, 0, \quad 10)^T$ directed to the origin. We rotated the camera randomly to generate a sequence of images. The number of shots was uniformly distributed

No. of images	Distance (m)	Relative errors (%)			
		$f_x^{(err)}$	$f_y^{(err)}$	$c_x^{(err)}$	$c_y^{(err)}$
10	2	0.5	5.3	3.6	-0.3
30	2	1.2	4.4	0.7	2
57	2	0.3	3.1	1.5	3.2
10	2	-1.8	0.7	-2.5	1.3
30	2	1.9	6	-0.3	8.6
74	2	0.1	4.3	0.2	7.6

Table 2: Relative errors for the auto-calibration of Logitech QuickCam Pro 900.

in the region $[3, 30]$. Camera position translation noise was generated uniformly in range $[\ \ t, t]$ where $t \in [0, 0.07]$ (distributed uniformly between experiments). Experimental results are shown in figure 9. It should be mentioned that in the case of high noise translation error the auto-calibration method is unstable and the method can end up with no solution. That's why we have less points on the figure when t is high.

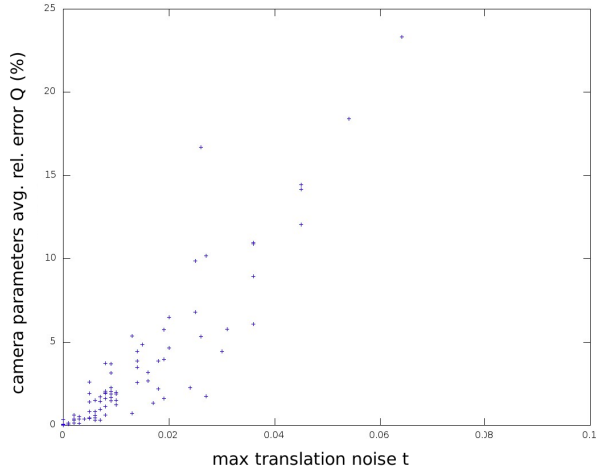

Figure 9: Translation noise effect on camera parameters estimation errors.

The chart implies that the camera parameters estimation errors are highly correlated with translational noise. A 3D reconstruction from images can be done only up to a scale, so instead of using absolute values in the chart we plot the ratio of the translation noise level to the distance from the camera to the nearest object in the scene.

From what we said above follows, that in practice one must try to take shots of distant objects or select images from source dataset where objects are distant. For distant objects homography registration will be more accurate, so in principle we can filter out close objects by high re-projection error.

According to the chart, one can say that in order to get a relative auto-calibration error about 10% it is recommended to make sure that the camera translation component divided by the distance to the objects is less than $\frac{0.025}{10}$ (according to the chart 0.025 translation noise corresponds to 10% level of auto-calibration error, where 10 is the distance from the camera to the sphere). That means about 2.5 cm shaking amplitude in case of 10 m distance must be ensured to get about 10% errors.

7. CONCLUSION

In our work we investigated the problem of auto-calibration for the case of rotational cameras and built a robust auto-calibration pipeline, which were tested successfully on real datasets.

We performed analysis of error in camera parameters impact on final results in such computer vision problem as image mosaicing, and showed that using modern stitching algorithms relaxes requirements on camera parameters accuracy, when theoretically errors in camera parameters can lead to big warping errors.

It is possible to calibrate cameras without patterns, but the main point is that the quality of input data is important for achieving accurate auto-calibration. For the case of rotational cameras auto-calibration it is necessary to ensure that translation noise relative to distance from camera to the nearest object is small enough.

8. REFERENCES

[1] Hartley, R. and Zisserman, A., "Multiple View Geometry in Computer Vision," Second Edition, Cambridge University Press, ISBN: 0521540518, 2004.

[2] M. Brown and D. G. Lowe, "Automatic Panoramic Image Stitching using Invariant Features," International Journal of Computer Vision, vol. 74, no. 1, pp. 59–73, 2007.

[3] Z. Zhang, "Flexible camera calibration by viewing a plane from unknown orientations," Proceedings of the 7th International Conference on Computer Vision, pp. 666673, Corfu, September 1999.

[4] Z. Zhang, "A flexible new technique for camera calibration," IEEE Transactions on Pattern Analysis and Machine Intelligence 22 (2000): 13301334.

[5] Gary Bradski, Adrian Kaehler, "Learning OpenCV: Computer Vision with the OpenCV Library," O'Reilly Media, 2008.

[6] Richard Szeliski, "Image Alignment and Stitching: A Tutorial," Microsoft Research, TechReport, MSR-TR-2004-92, 2004.

[7] Richard Szeliski and Heung-Yeung Shum, "Creating full view panoramic image mosaics and texture-mapped models," Association for Computing Machinery, Inc., Computer Graphics (SIGGRAPH'97 Proceedings).

[8] Ashley Eden, Matthew Uyttendaele, and Richard Szeliski, "Seamless Image Stitching of Scenes with Large Motions and Exposure Differences," IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'2006).

[9] M. Pollefeys, "Visual 3D Modeling from Images," Tutorial Notes, http://www.cs.unc.edu/ marc/tutorial/, 2000.

[10] D. Nister, "Untwisting a projective reconstruction," International Journal of Computer Vision, 2004.

[11] Simon Gibson, Jon Cook, Toby Howard, Roger Hubbold, Dan Oram, "Accurate Camera Calibration for Off-line, Video-Based Augmented Reality," ISMAR '02 Proceedings of the 1st International Symposium on Mixed and Augmented Reality.

[12] Jing Chen, Baozong Yuan, "Metric 3D reconstruction from uncalibrated unordered images with hierarchical merging," 2010 IEEE 10th International Conference on Signal Processing (ICSP)

[13] M.K. Chandraker, S. Agarwal, D.J. Kriegman and S. Belongie "Globally Optimal Algorithms for Stratified Autocalibration," IJCV 90(2):236-254, November 2010.

[14] Vivek Kwatra, Arno Schodl, Irfan Essa, Greg Turk, Aaron Bobick, "Graphcut Textures: Image and Video Synthesis Using Graph Cuts," To appear in Proc. ACM Transactions on Graphics, SIGGRAPH 2003.

[15] Adelson, E. H., and Burt, P. J. "Multi-resolution Splining Using a Pyramid Image Representation," SPIE Applications of Digital Image Processing IV, pp. 204-210, 1983.

[16] Bay, Herbert and Tuytelaars, Tinne and Van Gool, Luc "SURF: Speeded Up Robust Features," Computer Vision ECCV, Lecture Notes in Computer Science, Volume 3951/2006, 404-417, 2006.

[17] FLANN - Fast Library for Approximate Nearest Neighbors, http://people.cs.ubc.ca/ mariusm/index.php/FLANN/FLANN, 2011.

[18] OpenCV v2.4.0 stitching module documentation, http://opencv.itseez.com/modules/stitching/doc/stitching.html, 2012.

[19] Ji, Q., Dai, S., "Self-Calibration of a Rotating Camera with a Translational Offset," IEEE Trans on Robotics and Automation, V. 20, N.1, Feb 2004

[20] Junejo, I., Foroosh, H., "Practical PTZ Camera Calibration using Givens Rotations," IEEE ICIP 2008

ABOUT THE AUTHORS

Alexey Spizhevoy is a 2nd year M.Sc. student at Nizhny Novgorod State University and software engineer at Itseez where he created stitching and video stabilization modules in OpenCV library. He also ported pedestrian detection and optical flow estimation on GPU and integrated it into OpenCV GPU module. His contact email is `alexey.spizhevoy@itseez.com`.

Victor Eruhimov is chief technical officer of Itseez company. Prior to co-founding the company, he worked as a project manager and senior research scientist at Intel, where he applied computer-vision and machine-learning methods to automate Intel fabs and revolutionize data processing in semiconductor manufacturing. Before joining the manufacturing group he was developing technologies for human-motion capture, image retrieval, super-resolution, and face analysis. He is the author of more than 20 computer-vision and machine-learning papers, and holds several U.S. and international patents. His contact email is `victor.eruhimov@itseez.com`.

PANORAMAS

In this section we show panoramas that were obtained using OpenCV stitching pipeline [18] with camera parameters obtained with the described auto-calibration method.